2020-11

Blockchain Mergence and Reconditioning Blockchain to Enable Global Supply Chain Assurance

Hale, Britta; Brutzman, Don; Culbert, Jonathan; Norbraten, Terry

Monterey, California. Naval Postgraduate School

NPS-CS-21-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

BLOCKCHAIN MERGENCE AND RECONDITIONING

BLOCKCHAIN TO ENABLE GLOBAL SUPPLY CHAIN ASSURANCE

by

Britta Hale, Don Brutzman, Jonathan Culbert and Terry Norbraten

November 2020

Approved for public release: distribution unlimited

Prepared for: DCNO (Fleet Readiness & Logistics) (N4)

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From-To)*
14-01-2021	Technical Report	Apr 2020 – Jan 2021

4. TITLE AND SUBTITLE
Blockchain Mergence and Reconditioning Blockchain to Enable Global Supply Chain Assurance

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Hale, Britta, Ph.D., Brutzman, Don, Ph.D., Culbert, Jonathan and Norbraten, Terry

5d. PROJECT NUMBER
NPS-20-N362-B

5e. TASK NUMBER
W0A26

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Computer Science Department
Naval Postgraduate School
1 University Circle, Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CS-21-001

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DCNO (Fleet Readiness & Logistics) (N4)
Office of the Chief of Naval Operations
2000 Navy Pentagon
Washington, DC 20350-2000

Naval Research Program Management Office
Naval Postgraduate School
1 University Circle
Monterey, CA 93943

10. SPONSOR/MONITOR'S ACRONYM(S)
DCNO, NRP

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
The views expressed in this report are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

Distributed ledger technology such as blockchain could ideally be used to solve challenges in global supply chain assurance. In blockchain, consensus is achieved among active concurrent participants. The chain is, by design, required to be a single forward-building path of events; if branches appear, the chain consensus ensures that all but one branch is discarded. A supply chain in comparison, particularly on the production side, is a reversed architecture. In this case, small parts are used to build larger parts, hence requiring blockchain mergence (e.g., a final ready-for-use vehicle is comprised of multiple smaller parts sourced from various vendors, manufacturers, and even countries). Thus, the current capabilities of blockchain do not meet the fundamental demands of supply chains. Assuring supply chain integrity and visibility requires an adaptation of the technology to allow a form of blockchain mergence that the original concept was not designed to handle. This research looks at a possible solution among hash chains, blockchain, and ledger options for supply chain strategy.

15. SUBJECT TERMS
Blockchain Mergence, Distributed Ledger Technology (DLT), Hyperledger Fabric (HLF)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Britta Hale
Unclassified	Unclassified	Unclassified	UU	43	**19b. TELEPHONE NUMBER** *(include area code)* 831-656-3316 (DSN: 756)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau
President

Robert F. Dell
Acting Provost

The report entitled "Blockchain Mergence and Reconditioning Blockchain to Enable Global Supply Chain Assurance" was prepared for "DCNO (Fleet Readiness & Logistics) (N4)" and funded by the "Naval Research Program" (NRP).

Further distribution of all or part of this report is authorized.

This report was prepared by:

Britta Hale
Assistant Professor

Don Brutzman
Associate Professor

Jonathan Culbert
LCDR USN

Terry D. Norbraten
Faculty Research - Associate

Reviewed by:

Gurminder Singh, Chair
Computer Science

Released by:

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Distributed ledger technology such as blockchain could ideally be used to solve challenges in global supply chain assurance. In blockchain, consensus is achieved among active concurrent participants. The chain is, by design, required to be a single forward-building path of events; if branches appear, the chain consensus ensures that all but one branch is discarded. A supply chain in comparison, particularly on the production side, is a reversed architecture. In this case, small parts are used to build larger parts, hence requiring blockchain mergence (e.g., a final ready-for-use vehicle is comprised of multiple smaller parts sourced from various vendors, manufacturers, and even countries). Thus, the current capabilities of blockchain do not meet the fundamental demands of supply chains. Assuring supply chain integrity and visibility requires an adaptation of the technology to allow a form of blockchain mergence that the original concept was not designed to handle. This research looks at a possible solution among hash chains, blockchain, and ledger options for supply chain strategy.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AM	Additive Manufacturing
D-level	Depot Level
DLR	Depot Level Repairable
DLT	Distributed Ledger Technology
DoD	Department of Defense
ERP	Enterprise Resource Planning
GB	Gigabyte
GHz	Gigahertz
HLF	Hyperledger Fabric
IoT	Internet of Things
LCS	Littoral Combat Ship
MLS	Multilevel Security
O-level	Organizational Level
OSCM	Operations and Supply Chain Management
PLS-SEM	Partial Least Squares Structural Equation Modeling
QFD	Quality Function Deployment
RAM	Random Access Memory
RFI	Ready for Issue
SCM	Supply Chain Management
SOP	Standard Operating Procedures
TAM	Technology Acceptance Models
TPS	Transactions Per Second
TTP	Tactics Training and Procedures
UAV	Unmanned Aerial Vehicle
UTAUT	Unified Theory of Acceptance and Use of Technology

THIS PAGE INTENTIONALLY LEFT BLANK

I. BACKGROUND

Blockchain has been widely researched for applications due to the technology's ability to support consensus among distributed participants. The chain is, by design, required to be a single, forward path of events; if branches appear, the chain consensus ensures that all but one branch is discarded (Zheng et al., 2017). A supply chain, in comparison, particularly on the production side, is a reversed architecture. In this case, small parts are used to build larger parts, hence requiring a form of *mergence* (e.g., a final ready-for-use vehicle is comprised of multiple smaller parts sourced from various vendors, manufacturers, and even countries). Blockchain, according to its current design, fundamentally disallows this. Assuring supply chain integrity and visibility requires an adaptation of the technology to allow a form of blockchain mergence that the original concept was not designed to handle.

A. APPROACH

To survey existing blockchain solutions for forms of mergence, i.e., solutions for merging chains into a single blockchain, such as would be necessary for supply chain assurance.

To analyze potential solutions using partner signatures (where supply chain partners commit to chain addenda by digitally signing new blocks while also committing to the entire previous chain). This requires analysis of security considerations based on different commitment variants. Furthermore, it requires consideration of potential timelines and timeline collisions of block production.

The above solutions are evaluated with respect to formal blockchain integration. In particular, the research investigates whether or not mergence of distributed ledgers is possible within exiting blockchain architectures, or if it is feasible as a parallel assurance mechanism, such that commitments are uploaded to an existing blockchain. This evaluation will be made on mathematically feasibility as well as use case comparison.

II. CURRENT RESEARCH ON LEDGER MERGENCE

Blockchain technology has been often touted as solution to various challenges since its inception under Bitcoin and cybercurrency. However, blockchain technology may benefit Navy logistics. In essence, blockchains are a list of records, or blocks, cryptographically linked as a distributed ledger for recording transactions among parties in a permanent and verifiable way (Zheng et al., 2017). Blockchain could also support "smart contracts" which may be away to reduce administrative friction.

The hallmarks of a robust Distributed Ledger Technology (DLT) are decentralization between blockchain networks and the individual nodes in those networks, as well as the consensus reached when validating individual blocks when added to a network's blockchain ledger (Khan, 2019). Khan (2019) also notes that characteristics such as the number of transactions per second (TPS) that a network can process, the network's scalability, and how a particular network guards against malicious attempts to add false information, are also key to a good system.

This work focuses on authentication of changes at the micro-level, with transparency in a ledger for support of supply chain assurance. Industry is working on a number of efforts involving supply chain logistics and supply chain management such as Hyperledger (2020), Everledger (2020), and Ethereum (2020) that may have an application to the Navy's logistical systems and perhaps could contribute to an agile logistical system. The central challenge is applying such efforts beyond acquisitions to the whole lifecycle of the supply chain.

A. A QUICK LOOK AT DLT THEN AND NOW

Nakamoto (2009) is considered the conventional originator of the original description of blockchain technology, although it is focused on the financial and Bitcoin applications. One should note, however, that the concepts surrounding blockchain predate this by a decade or so, and there is other research available on distributed ledgers previous to that timeframe. Beyond Bitcoin, DLT and blockchain have been researched for various financial and operational tracking purposes. Zheng et al. (2017) and Natarajan

et al. (2017) provide a general and fairly informal introduction into DLT and how it might integrate into mainstream day to day operations in the financial, private, and government sectors. Natarajan et al. (2017) also provides a sense of how "decentralized records of flow of commodities and materials across a supply chain by using trusted stakeholders to validate flows and movements" could benefit those stakeholders, leading credence to adopting DLT which would enhance trust in the supply chain. For an overview of blockchain research, consult Fosso et al., (2020), which highlights the benefits of the creation of value in operations and supply chain management (OSCM). Statistics such as the number of published papers by country, topic, keyword summary and relationships are recorded.

Although not explicitly addressing blockchain technology, Bonanni (2011) discusses supply chain discovery/awareness, concepts and concerns that motivate the current work. Bonanni argues for "Radical Transparency" in the context of sustainable (carbon cost) supply chains, carbon-footprint measured supply chains and product life cycle awareness and optimization. This runs into a similar problem set that DoD acquisition may encounter – companies' unwillingness to reveal their supply chain details as trade secrets, or an inability to do so, being unaware of the source of their sources.

There has also been a line of research covering direct application of blockchain to supply chain management. Korpela et al. (2018) provides an analysis of how blockchain could be used to solve or ameliorate the issues of concern of the major stakeholders involved in a very large supply chain operation. The main contribution of the paper is proposed elimination of a third party to mediate/handle supply chain inter-business and then address these popular concerns as graphed in the following Table 1. Meanwhile, Banerjee (2018) provides an overview/summary of the use and benefits of blockchain in supply chain operations such as:

- Reduced counterfeiting and origin tracing
- Digital product details/lifecycle
- Custom-built provenance solutions: Software service providers can use the blockchain framework to build provenance solutions for its customers (permission blockchain)

Based on Banerjee's work, custom-built solutions appear to have gained traction within industry. For instance, Infosys® has developed a product provenance solution using Oracle® Blockchain Cloud Services that is based on Hyperledger™ Fabric. Infosys® has also developed a coffee bean tracking provenance solution for its customers. Such examples point towards a demand for custom built provenance solutions that can be developed with product or industry specific validations. It is important to note that the concept of provenance only functions when all the supply chain stakeholders are part of the blockchain network. The architecture of blockchain inherently traces products as they pass from one supply chain entity to another. These transactions are stored as blocks and chronologically linked according to the physical movement of "the goods." Supporting such tracking technologies motivates our solution (see Section C).

Kshetri (2018) provides a theoretical framework related to key objectives of Supply Chain Management (SCM). Kshetri's work covers several corporate case studies of how the Internet of Things (IoT) blockchain SCM can be used by companies with differing levels and areas of interest in supply chain verification/source confidence (see Table 1). Such case studies, including the Chipotle™ E-coli outbreak ingredient tracing case study, may shed light on potential parallel solution behaviors involving a faulty/compromised hardware component recall in the DoD. Under a similar formal goal, Queiroz et al. (2018) covers blockchain SCM adoption in the U.S. and India. The study advocates for drawing on emerging literature on blockchain, supply chain and network theory, as well as on technology acceptance models (TAMs). Queiroz et al. (2018) introduce a model based on a slightly altered version of the classical unified theory of acceptance and the use of technology (UTAUT).

Table 1
The cases selected and their classification in terms of incorporation of the IoT and deployment of blockchain to validate individuals' and assets' identities.

Deg. of incorporation of IoT → / Deg. of deployment of blockchain to validate identities of individuals and assets ↓	High	Low
High	Maersk	Lockheed Martin Everledger
Low	Alibaba Chronicled Modum Walmart Gemalto Intel's solution to track seafood supply chain	Bext360 Provenance

Table 1. Cases selected and their classification in terms of incorporation of the IoT and deployment of blockchain to validate individuals' and assets' identities. (Kshetri, 2018)

III. A METHOD FOR ADAPTING DISTRIBUTED LEDGER FOR SUPPLY CHAIN USE

A. UAV USE CASE: SYSTEMS DEPLOYED BY U.S. NAVY SHIPS

Defining unique supply-chain characteristics of deployed Naval assets increasingly depend upon the deployment, maintenance and just-in-time improvement of unmanned systems.

Littoral Combat Ships (LCS) have two classes of relatively small surface warships designed for operations near shore by the U.S. Navy ("Littoral combat ship," 2020). Modern designs allow for flexible mission execution, various mission payloads, and other tasking. Reduced crew complements mean individuals are assigned, yet with reduced inventories of spare parts and supplies.

The use of unmanned aerial vehicles (UAVs) help in this regard. These vehicles are employed for scouting and other rapid response detailing that minimize risk to the overall mission, ship and crew. The ecosystem for a typical UAV consists of four categories of components:

- Hardware: airframe, sensors, computers
- Software: communication, guidance and control
- Additive Manufacturing (AM): 3D printed wings, tails and other small parts for ad hoc repair
- Information: keys, training, repair instructions, feedback, safety

Within these four categories of components, each is different and necessary for aggregation into a complete device, and each has different stakeholders and supply chains feeding ships supplies. Thus, four parallel supply chains of interest exist, and each is interdependent; therefore, any mergence solution should necessarily support all four aspects, as seen in Figure 1. Note that even with acquisition of a device as a single unit, the nature of updates, potential repairs, and parts reuse between devices imply that it must be possible, for tracking purposes over the device lifetime, to handle the mergence of all four aspects.

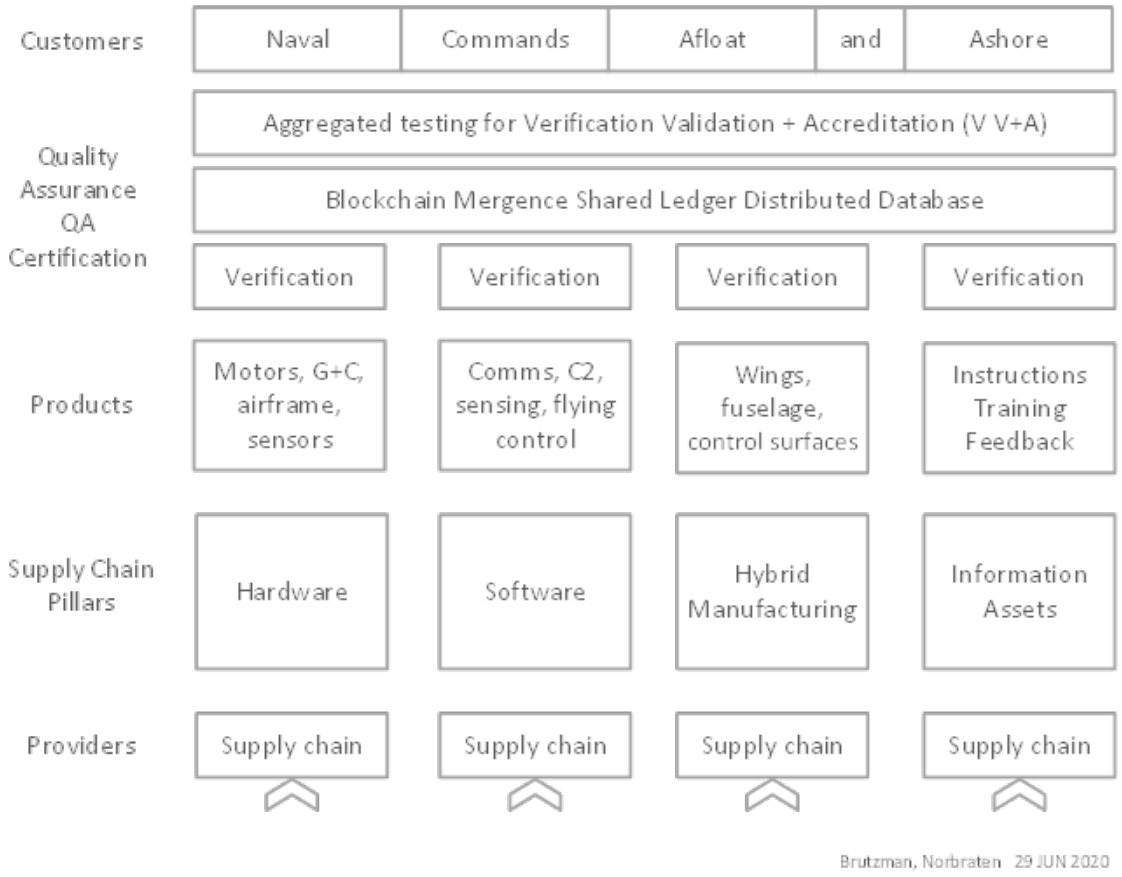

Figure 1. UAV Operational Assembly and Modification

B. SYSTEM CONSTRAINTS AND REQUIREMENTS IDENTIFICATION

In this section the various system requirements are explored, in the context of the UAV use case.

1. Scenario: UAV Deployment, Repair, Operations

Suppose that a ship deploys with stock gear and consists of two distinct, yet similar versions of a UAV. The ship must maintain its current pace of operations until return to port, or resupply.

Under normal operations, the following issues may affect device history, in that they impact the integrity of the device or its trustworthiness, and therefore should be added in an authenticated manner to the device history:

- Software updates

- Training and safety updated to ship standard operating procedures (SOP) and tactics, training and procedures (TTP)

Now suppose that a collision occurs during testing between the two UAV causing damage to each vehicle. The following may also be important changes to the device history, requiring authenticated changes in device records:

- Hardware replacements on board, to include classified components
- 3D printing for upgraded tail assemblies
- Maintenance feedback to shore commands

Any mergence solution must therefore support, per minimum such a variety of changes to the item history.

2. Mergence Requirements Assessment – Verifiability

In addition to the afore mentioned types of item record changes, there are also requirements in how a change is recorded. In particular, the record must be verifiable. In terms of verifiability, the following requirements are also essential and must be supported by a mergence solution:

- Conformation that a given component X is on the ship
- Conformation of all devices in the inventory that have X as a component
- Conformation if and when X has been replace/repaired/etc., within a particular device
- Conformation of the change entity – the responsible party to change/split/remove/combine X as a component within devices
- Ability to add logs or metadata

The above requirements emerge from use case issues. For example, if a device component is found to be compromised and must be removed, the logged data associated with the device should indicate if it has been removed and by whom. Furthermore, is it important for administration purposes to identify all possible devices containing the compromised component for swift handling and damage mitigation. In these contexts, “components” may refer not only to hardware, but also malicious software or poorly executed additive manufacturing (e.g., 3D printed wings with vulnerable integrity).

3. Mergence Requirements Assessment – Flexibility

Finally, flexibility requirements associated with mergence are listed. Since mergence solutions must support potential external (industry/non-DoD) supply chain tracking of unpredictable natures, the mergence solution must be fairly adaptable. Furthermore, external supply chain tracking may differ from internal (DoD) tracking, and the potential solution must support one or more blockchains used internally to the DoD. As many acquisition devices may be of a sensitive nature, the mergence solution must furthermore support various classification levels, such that unclassified devices may be administered in unclassified environments, while devices of higher classification levels can also be managed within the same mergence solution without sensitive information leakage. Finally, in addition to all of these, devices transfer hands between organizations ships, etc., requiring a flexibility to record management. This leads us to the following final four solution requirements:

- Flexibility independent of source/industry in the external supply chain
- Flexibility with internal blockchains(s) within DoD
- Flexibility with classification levels
- Flexibility for device transfer between organizations/ships/etc. internally

C. LEDGER MERGENCE EITHER IN BLOCKCHAIN OR AS A MODULAR APPROACH LEVERAGING EXISTING BLOCKCHAIN SOLUTIONS

There is a natural separation between external-DoD and internal-DoD supply chain tracking. This intrinsically leads to a dual solution, with the acquisition boundary denoting a change in authenticity tracking. Even for internal supply chain tracking, satisfying all solution requirements appears, on the outset, to be impossible. Notably a solution that crosses classification boundaries must be carefully handled, especially for full item records and tracking information. This is handled by further separating out the internal DoD authentication chain into two parts.

1. External and Internal Chains

DoD equipment is typically procured via outside commercial manufacturing vendors. The supply chain starts outside of the DoD where parts and other equipment must be verified and validated before becoming available inside internal supply chains. Conceptually, manufacturers many require supply chain assurance as well, tracking purchased components for integration in building devices. This may take the form of various blockchains (see Figure 2 "Supply chain"). Minimally, manufacturers may be required to present verification on the types and sources of a device's components. At acquisition, a new item record will be formed, such that the component history of the acquired device is verified and authenticated by the acquisition authority, who registers components under a digitally signed genesis block. Once a genesis block for the internal ledger is formed, tracking may proceed internally.

What is essential at the DoD boundary/component registration step is that actual verification of internal components to a device. Information on processing chips, software, extra must be recorded. This enables future tracking such that if, for instance, a component is later discovered to be compromised in the manufacturing chain, all devices containing the critical component can be identified. The genesis block thus serves as an initial registration for all components, such that it is only necessary to record changes to that initial list within the device history record.

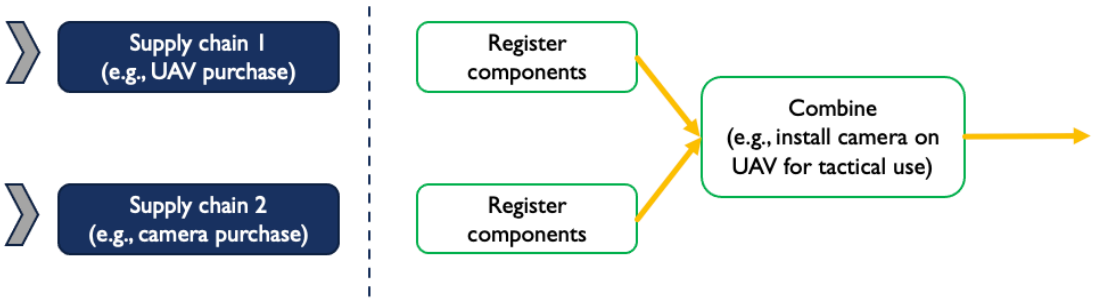

Figure 2. DoD Acquisition: Verification of External Input Chains(s), Registering of Devices to Initiate Internal Chains

2. Mergence Operations

Internal tracking is broken into two further chains to support classification boundaries. The *device chain* handles immediate time history, and authenticates records

as visible to the admin. Moreover, the device chain is designed to support the current Depot Level Repairable (DLR) system. Within the mergence solution, in addition to the device chain, one or more internal blockchains are also supported. These may be organization level blockchains, or classification level blockchains. Here, the term blockchain is employed for a distributed immutable ledger, without specification that restricts to any particular ledger format or consensus method. This in turn meets the flexibility requirements specified in Section B.

The following device chain operations are fixed, in accordance with the requirements:

- *Device registration*: adding new, original item to a ledger. This creates the genesis block for the device chain.
- *Device repair*: adding a new component onto an existing device. This differs from *device combine* in that the component being added has no registration history (i.e., no genesis block). This may occur if the repair takes place using additive manufacturing.
- *Device split*: separation of components within an existing device. This supports potential re-use or disposal, such as when a component breaks and is removed from the current item record (device history is still maintained). This creates two separate device chains: one for each split component.
- *Device combine*: integrate two components into a new combined device. This supports customization of devices after acquisition and parts replacement (e.g., a newly purchased component added to an existing device).

Device split can be employed if a device breaks, but components can be reused. For example, suppose that a UAV (UAV1) malfunctions but certain components can be used to repair another UAV (UAV2). The broken device would then have a device split operation in its item record, creating two new chains: one for component that will be reused and one for the remaining unusable assembly UAV1. A device combine operation then integrates the split component into UAV2. As such, the item history of UAV1 is

now linked to UAV2. If there were relevant repairs to the reused component or if it comes to light that the reused component was compromised during manufacture and must be pulled from use, it will be immediately clear from UAV2's record history that the part now resides within UAV2 instead of the UAV1 device carcass.

Each of the stated operations must be authenticated. For this be use the PKI infrastructure already inherent in the DLR system. The operator responsible for the device signs the various operations. The signature covers the current record for the device(s) being operated on as well as what type of operation is performed. The authenticated transcript is stored as part of the device chain. These operations are shown in Figure 3.

The distributed ledger and shared memory exist beyond the immediate device chain history, such that an item record cannot be changed *a posteriori*. For this, a blockchain is employed, which records the signatures from the device chain operations. Note that only signatures are required, and not the related device information, to be stored on the blockchain, although the latter may be. Storage of further information or metadata may be beneficial for device tracking but could also leak information (such as if the device or its location is sensitive). Instead, only the minimum information on the blockchain concerning the current signature state is required.

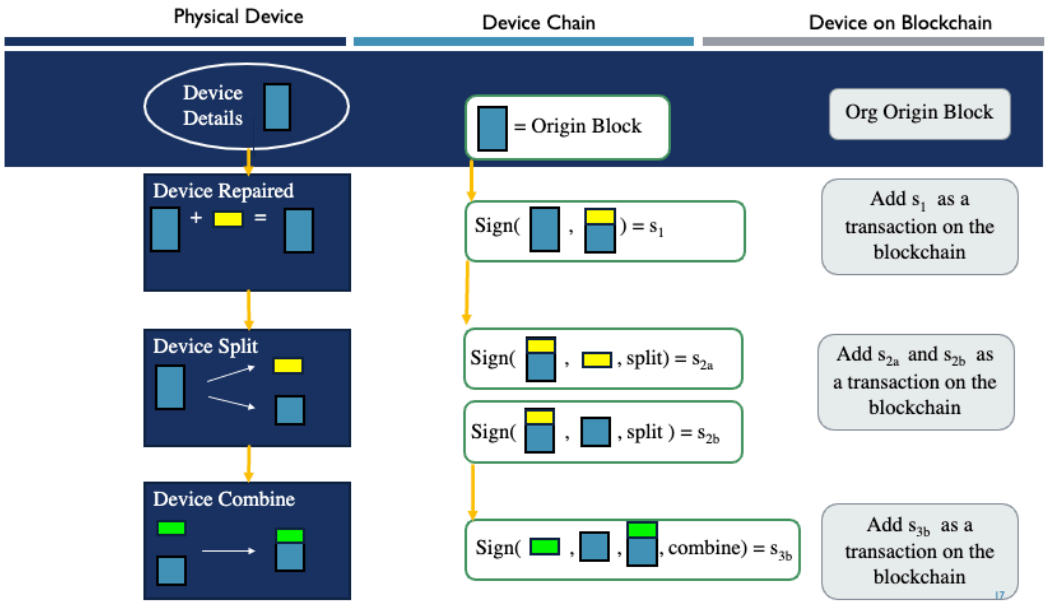

Figure 3. Sequencing Operations for Blockchains

3. Multilevel Security (MLS) Classification Agility Considerations

For hybrid devices used in the fleet, activities may occur and be needed across multiple levels and domains of security, such as: UNCLASS, CONFIDENTIAL, SECRET, and TOP SECRET. A solution is to map to a Multilevel Security (MLS) classification system and demonstrate interoperability.

Note that while the device chain contains potentially classified information, the information sent to the blockchain is comprised of merely the signature on the data vs. the data itself; any further additions are optional. Even with a time code associated to the signature object representation there is no intrinsic value to the information outside of the context of the signed data, especially with a plentitude of blockchain transactions. Thus, the blockchain information can be shared across multi-level security systems since these codes are useless without ledger/database access.

In addition to the above observation, blockchains may be allowed to operate at different classification levels, such that more relevant device information may in fact be added to the blockchain than merely the signature. This in turn implies that any device may have a record with varying classification levels attached to different aspects of the associated information, and that the associated data may be placed on the relevant blockchain. Naturally, higher classification can correlate same and lower-level data records, but not write to them, per the properties of the MLS system. Figure 4 illustrates this framework in practice.

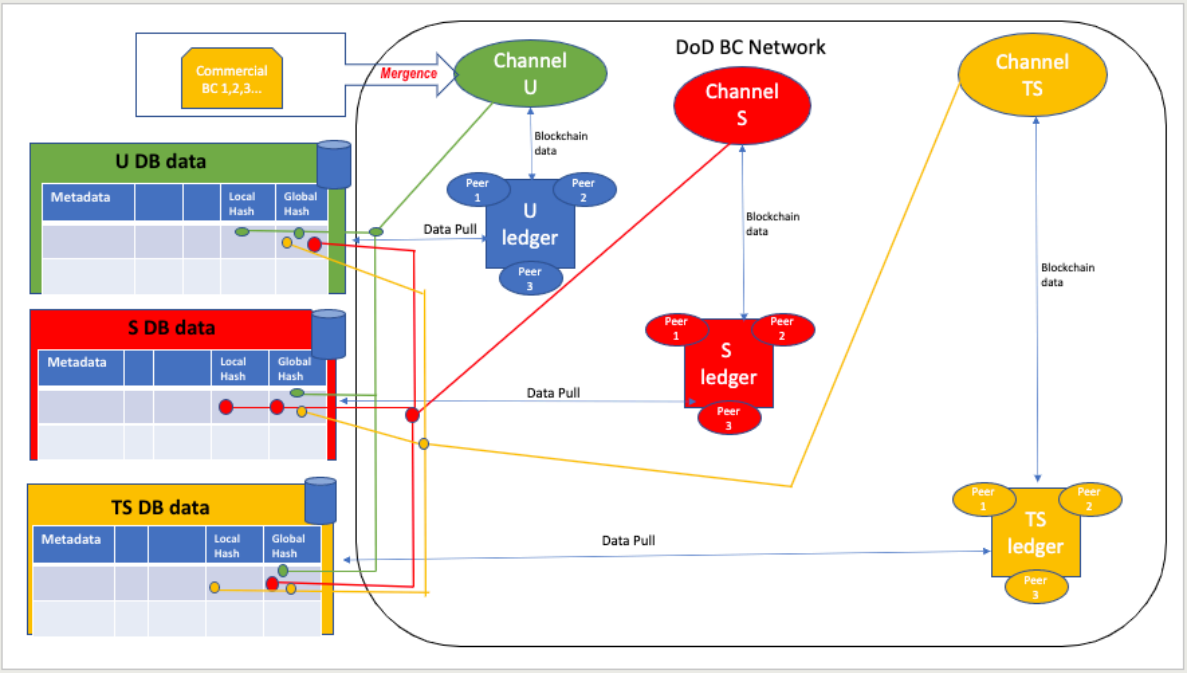

Figure 4. Multilevel Security (MLS) Classification Agility within a DoD Blockchain Network

To consider how this may work in practice, the reader can walk through the following conceptualized steps for handling the mergence solution of device chains and blockchains within MLS:

1) A new item genesis record for an UNCLASS device is created for the device chain. This correlates to a device origin record with signature information populated to the various blockchains.
2) The device is transferred from an UNCLASS environment to a SECRET environment. The device chain is now maintained at SECRET.
3) The device is repaired, using a combine operation on the device chain. Necessary information is populated to the appropriate and corresponding SECRET level blockchain, while other chains record signature information only.

If it is later discovered that the device contained a malfunctioning or compromised component (e.g., through manufacturer notification), then an operator can

see all associated components in the device genesis block, recognizing that the critical component is present. It can then be seen that the device as associated to a different classification (or organization blockchain) and the appropriate authority for that blockchain can be contacted, who can then trace the device's history to identify if the component is still present or has been replaced.

Strengthening Naval supply-chain accountability, integrity and trust that is distributed across exceptionally ling, intermittent and diverse communication networks is an example of Data-Centric Security, which is an important topic being explored as part of the Network Optional Warfare (NOW) project (Network Optional Warfare, 2020).

D. DOD EQUIPMENT REPAIR TRANSACTIONS USING THE HYPERLEDGER™ FABRIC FRAMEWORK FOR DISTRIBUTED BLOCKCHAIN LEDGER KEEPING

It is important to show that such theoretical functionality is indeed achievable in practice. This project demonstrates and end-to-end exemplar scenario using the industry-grade software, the open-sourced Hyperledger™ Fabric (HLR) Framework (Hyperledger, 2020) hosted on GitHub (2020). This exemplar has been constructed to showcase what various repair level organizations might use to record supply chain transactions between them using DLT to merge separate blockchains into one.

1. UAV Camera

Present in the exemplar scenario are an organizational level (O-level) end user of an unmanned aerial vehicle (UAV) in its custody that houses a camera subcomponent, a depot level repairable (DLR) that requires repair at the depot level (D-level), and the D-level repair station. The client application transactions that take place and recorded on the blockchain ledger are:

1) O-level issues the non-functional DLR camera to D-level for repair
2) D-level accepts and conducts the required repairs for the DLR camera
3) D-level reissues the repaired DLR camera back to O-level

The intent is to show chain of custody for the DLR camera subcomponent, camera metadata, i.e., serial number, and status of repair of the DLR in the supply chain and how each O-level and D-level ledger merge to comprise the DLR repair history.

The blockchain network, N, will comprise of the following consortium organizations, components and entities (see Figure 5):

- Organizations R1 (D-level), R2 (O-level) & R4 (blockchain network administrator).
- Client applications A1 (D-level transactor) & A2 (O-level transactor). Client applications conduct transactions on behalf of their respective organizations.
- Certificate authorities CA1, CA2 & CA4. Each organization can prefer their own vetted certificate authority.
- Peers P1 (D-level) & P2 (O-level). Peers maintain local copies of and record blockchain ledger transactions in accordance with agreed upon smart contracts (chaincode) within the consortium.
- Blockchain ledger L1. Each peer maintains and communicates with other network peers to ensure local blockchain ledger copies are kept uniform throughout the network.
- Smart contract (chaincode) S5. Peers are able to maintain blockchain ledger uniformity through consortium member agreed upon smart contracts.
- Network ordering service O4. The ordering service serves as the initial administrative gateway between consortium members upon network standup.
- Network configuration NC4. Consortium members R1, R2 and R4 all agree upon the blockchain network configuration policies administered by ordering service O4 via NC4.
- Channel configuration CC1. The channel configuration allows for network peers to accept and distribute blockchain ledger transactions between authorized organizations in accordance with NC4.
- Channel 1. The communications channel where organizational peers accept and record transactions between client applications A1 & A2 on channel 1.

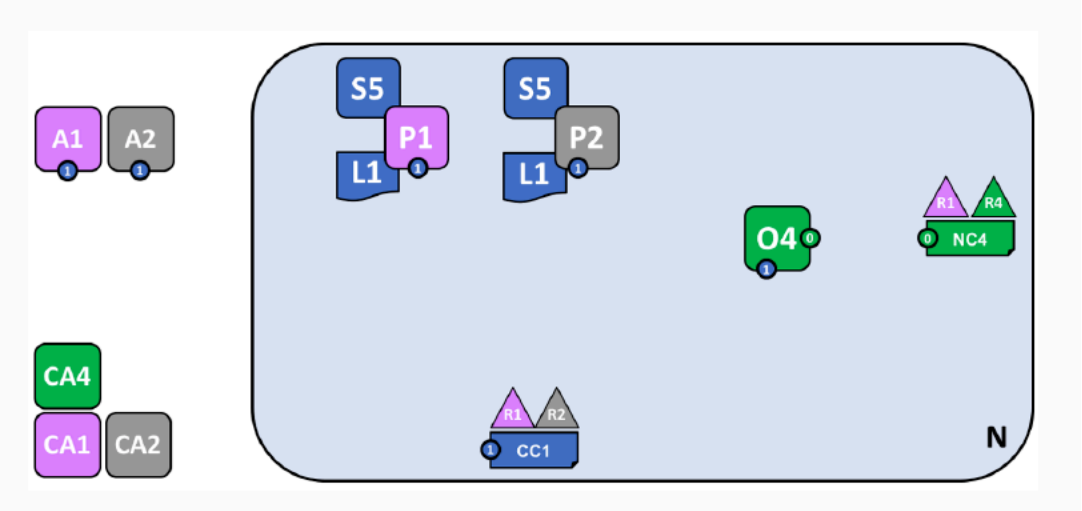

Figure 5. DoD Equipment Repair Blockchain Network (N) Configuration Topology

2. Using the Fabric Test Network

Hyperledger (2020) provides a test network that developers can download and work their way through a demonstration of the capabilities of Hyperledger™ Fabric. The portal for that demonstration is here: https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html. Depending on the operating system of the developer's choice, a specific environment is first set up on a local machine in order to leverage the Hyperledger™ Fabric framework. Once the prerequisites are complete, the test network is invoked via command line scripts and the demonstration can proceed. In our case the demonstration code was modified locally to mimic our UAV camera scenario.

3. DLR Transactions in Action

The following figures are from screenshots of a live network (N) demonstration.

Upon network standup/startup, the various organizational peers, admins, etc. are defined, registered, enrolled and assigned certificate authorities which then issue authenticating certificates for each respective network node. The agreed upon smart contract (chaincode) is deployed to each organizational peer node and tested. Finally, each organizational peer node is given local custody of the blockchain ledger which is then readied for acceptance of and recording of ledger transactions (see Figures 6-9).

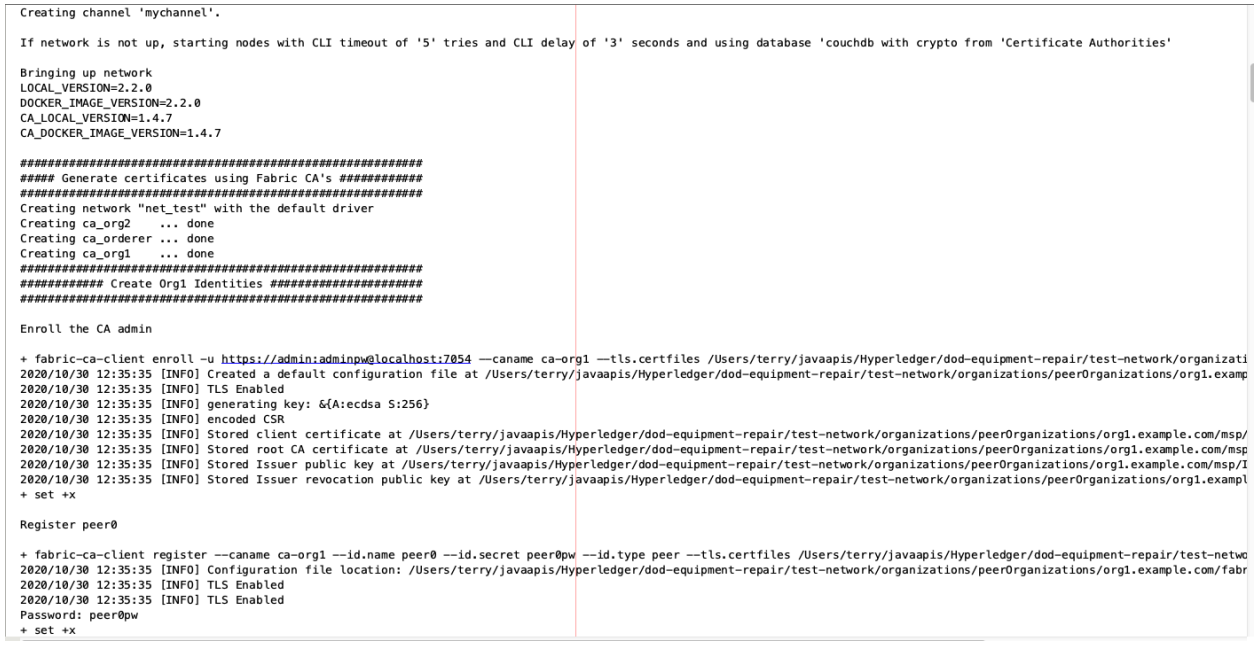

Figure 6. Registering and Authentication of All Network Nodes upon Network (N) Standup

```
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
WARNING: Found orphan containers (ca_org2, ca_org1, ca_orderer) for this project. If you removed or renamed this service in your compose file, you can run this command
Creating couchdb0              ... done
Creating orderer.example.com   ... done
Creating couchdb1            ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS                  PORTS
c5219eab2f72        hyperledger/fabric-peer:latest      "peer node start"        1 second ago        Up Less than a second   0.0.0.0:7051->7051/tcp
4a8c4c566b59        hyperledger/fabric-peer:latest      "peer node start"        2 seconds ago       Up Less than a second   7051/tcp, 0.0.0.0:9051->9051/tcp
185cc83f6c4c        couchdb:3.1                         "tini -- /docker-ent…"   3 seconds ago       Up 1 second             4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp
145a21411d18        couchdb:3.1                         "tini -- /docker-ent…"   3 seconds ago       Up 1 second             4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
c731d0d7b4e1        hyperledger/fabric-orderer:latest   "orderer"                3 seconds ago       Up 1 second             0.0.0.0:7050->7050/tcp
f5f553e33369        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   20 seconds ago      Up 19 seconds           0.0.0.0:7054->7054/tcp
2d0467fd386f        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   20 seconds ago      Up 19 seconds           7054/tcp, 0.0.0.0:9054->9054/tcp
3531efb52757        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   20 seconds ago      Up 19 seconds           7054/tcp, 0.0.0.0:8054->8054/tcp
### Generating channel create transaction 'mychannel.tx' ###
+ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/mychannel.tx -channelID mychannel
2020-10-30 12:35:44.578 PDT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-10-30 12:35:44.621 PDT [common.tools.configtxgen] main -> INFO 002 Loaded configuration: /Users/terry/javaapis/Hyperledger/dod-equipment-repair/test-ne
2020-10-30 12:35:44.621 PDT [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx
2020-10-30 12:35:44.632 PDT [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
+ res=0
+ set +x

### Generating anchor peer update transactions ###
#######    Generating anchor peer update transaction for Org1MSP  ##########
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID mychannel -asOrg Org1MSP
2020-10-30 12:35:44.696 PDT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-10-30 12:35:44.737 PDT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /Users/terry/javaapis/Hyperledger/dod-equipment-repair/test-ne
2020-10-30 12:35:44.737 PDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-10-30 12:35:44.744 PDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
+ res=0
+ set +x

#######    Generating anchor peer update transaction for Org2MSP  ##########
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID mychannel -asOrg Org2MSP
2020-10-30 12:35:44.815 PDT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-10-30 12:35:44.854 PDT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /Users/terry/javaapis/Hyperledger/dod-equipment-repair/test-ne
2020-10-30 12:35:44.854 PDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-10-30 12:35:44.867 PDT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
+ res=0
+ set +x
```

Figure 7. Network Channel Configuration and Standup of Organizational Peers

```
# deploy chaincode
pushd "${DIR}/../test-network/"
~/javaapis/Hyperledger/dod-equipment-repair/test-network ~/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle
./network.sh deployCC -l ${CC_SRC_LANGUAGE}
deploying chaincode on channel 'mychannel'

Compiling Java code ...
~/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle/chaincode/contract-java ~/javaapis/Hyperledger/dod-equipment-repair/test-network
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

BUILD SUCCESSFUL in 11s
3 actionable tasks: 3 executed
~/javaapis/Hyperledger/dod-equipment-repair/test-network
Finished compiling Java code
Using organization 1
++ peer lifecycle chaincode package cp.tar.gz --path ../unmanned-aerial-vehicle/chaincode/contract-java/build/install/uavcontract --lang java --label uavcontract_1
++ res=0
++ set +x
==================== Chaincode is packaged on peer0.org1 ====================

Installing chaincode on peer0.org1...
Using organization 1
++ peer lifecycle chaincode install cp.tar.gz
++ res=0
++ set +x
2020-10-30 12:36:29.825 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nNuavcontract_1:4052ecc3d0172dcbd17e61
2020-10-30 12:36:29.826 PDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: uavcontract_1:4052ecc3d0172dcbd17e6137ab32bc772e245ae6
==================== Chaincode is installed on peer0.org1 ====================
```

Figure 8. Chaincode (Smart Contract) Deployment to Each Organizational Peer

```
Using organization 1
Using organization 2
++ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /Users/terry/javaapis/Hyperledger/dod-equipment-repair/test-network/or
++ res=0
++ set +x
2020-10-30 12:37:10.711 PDT [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
==================== Invoke transaction successful on peer0.org1 peer0.org2 on channel 'mychannel' ====================

Querying chaincode on peer0.org1...
Using organization 1
==================== Querying on peer0.org1 on channel 'mychannel'... ====================
Attempting to Query peer0.org1, Retry after 3 seconds.
++ peer chaincode query -C mychannel -n uavcontract -c '{"Args":["org.hyperledger.fabric:GetMetadata"]}'
++ res=0
++ set +x

{"components":{"schemas":{"UavCamera":{"additionalProperties":false,"type":"object","properties":{"operationalStatus":{"type":"string"},"owner":{"type":"string"},"cameraNumber":
==================== Query successful on peer0.org1 on channel 'mychannel' ====================
```

Figure 9. Schema Checking and Transaction Testing of Deployed Chaincode Between Organizational Peers

The network (N) is now ready to facilitate transactions. Each client application (authorized organizational transaction entity) then submits their authentication data to network (N) before the network authorizes transactions to take place.

```
cd /Users/terry/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle/consortium/dlevel/application-java; JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Cor
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their
Scanning for projects...

-----------------------< uav-camera-repair:d-level >------------------------
Building Depot Level Repair 0.0.1-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:1.5.0:exec (default-cli) @ d-level ---
credentialPath: ../../../../test-network/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp
certificatePem: ../../../../test-network/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp/signcerts/User1@org1.example.com-cert.pem
Write wallet info into ./wallet successfully.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  9.206 s
Finished at: 2020-10-30T17:05:04-07:00
------------------------------------------------------------------------
```

Figure 10. Authentication of Client Application Entities (Pre-transactional)

Finally, the network (N) is operational, and authorized organizational entities are recognized. Transactions may begin now, but only those which are explicity defined in the smart contract. Figure 11 beings with an O-level entity submitting a UAV camera to D-level for repair.

```
cd /Users/terry/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle/consortium/olevel/application-java; JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Cor
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their
Scanning for projects...

-----------------------< uav-camera-repair:o-level >-----------------------
Building Oranizational Level Repair 0.0.1-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:1.5.0:exec (default-cli) @ o-level ---
Read wallet info from: ./wallet
Use network channel: mychannel.
Use org.repairnet.uavcamera smart contract.
Submit uav camera issue transaction.
Process issue transaction response:
Camera: 00001
Issuer: O-Level
Operational status: not functional
State: successfully ISSUED
Owner: O-Level
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  30.731 s
Finished at: 2020-10-30T17:03:52-07:00
------------------------------------------------------------------------
```

Figure 11. O-level UAV Camera Issue Transaction

```
cd /Users/terry/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle/consortium/dlevel/application-java; JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Cor
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their
Scanning for projects...

-----------------------< uav-camera-repair:d-level >-----------------------
Building Depot Level Repair 0.0.1-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:1.5.0:exec (default-cli) @ d-level ---
Read wallet info from: ./wallet
Use network channel: mychannel.
Use org.repairnet.uavcamera smart contract.
Submit uav camera repair transaction.
Process repair transaction response:
Camera: 00001
Issuer: O-Level
Operational status: in repairs
State: successfully REPAIRING
Owner: D-Level
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  26.853 s
Finished at: 2020-10-30T17:06:07-07:00
------------------------------------------------------------------------
```

Figure 12. D-level UAV Camera Repair Transaction

```
cd /Users/terry/javaapis/Hyperledger/dod-equipment-repair/unmanned-aerial-vehicle/consortium/dlevel/application-java; JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Cor
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their
Scanning for projects...

-----------------------< uav-camera-repair:d-level >-----------------------
Building Depot Level Repair 0.0.1-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:1.5.0:exec (default-cli) @ d-level ---
Read wallet info from: ./wallet
Use network channel: mychannel.
Use org.repairnet.uavcamera smart contract.
Submit uav camera reissue transaction.
Process reissue transaction response:
Camera: 00001
Issuer: O-Level
Operational status: ready to reissue
State: successfully REISSUED
Owner: O-Level
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  23.962 s
Finished at: 2020-10-30T17:07:02-07:00
------------------------------------------------------------------------
```

Figure 13. D-level UAV Camera Reissue Transaction

Figures 11-13 show the three transactions completed, as per the smart contract constructs initiated from each of the O-level and D-level authorized entities. These transactions were initiated from each client gateway interface application that has knowledge of the network (N) from their respective remote locations. The next two figures 14-15 show what the network logs as the transactions progress from the D-level perspective.

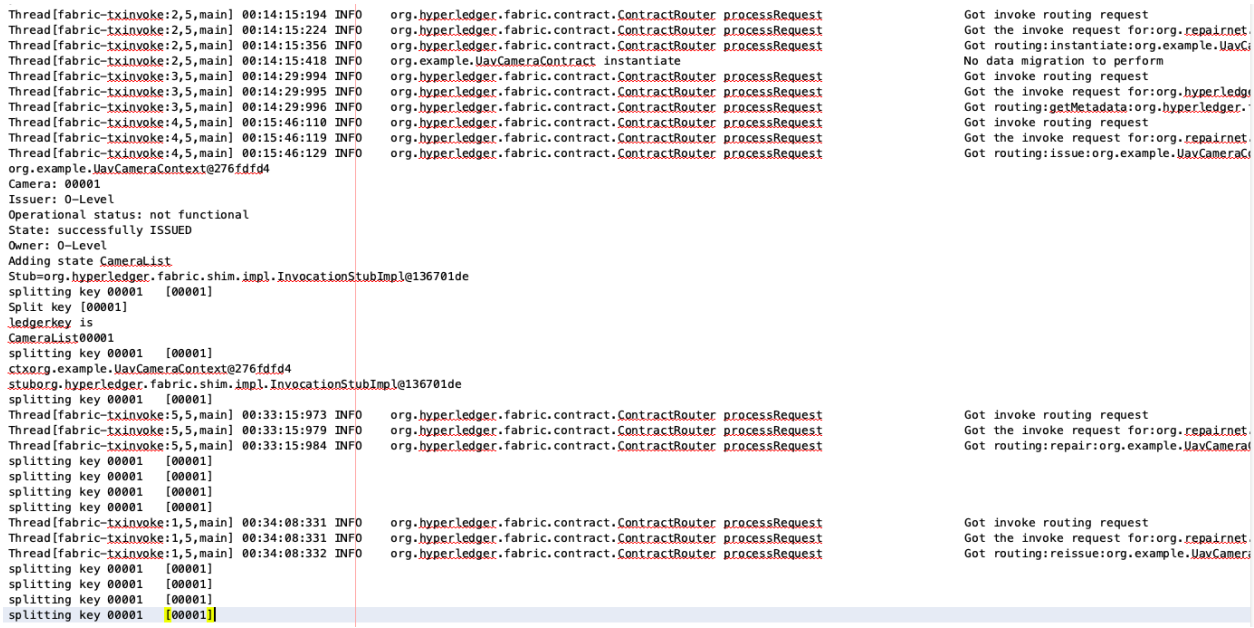

Figure 14. D-level Peer Showing Chaincode Testing and All Client Application Transactions Received and Processed

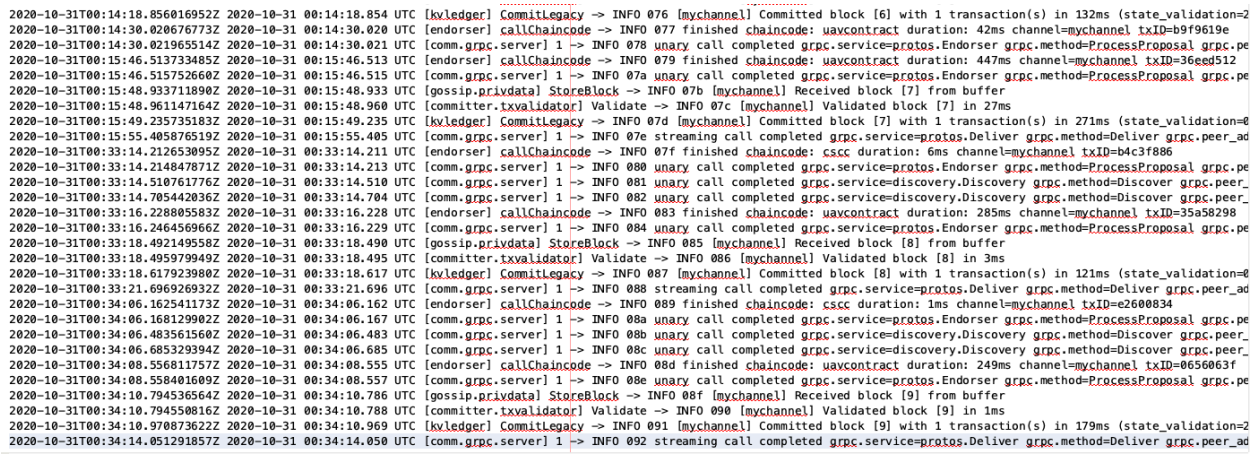

Figure 15. D-level Peer Showing Blockchain Ledger Transaction Commits

```
./network.sh down
Stopping network

Stopping peer0.org2.example.com ... done
Stopping peer0.org1.example.com ... done
Stopping couchdb0               ... done
Stopping couchdb1               ... done
Stopping orderer.example.com    ... done
Stopping ca_org2                ... done
Stopping ca_orderer             ... done
Stopping ca_org1                ... done
Removing peer0.org2.example.com ... done
Removing peer0.org1.example.com ... done
Removing couchdb0               ... done
Removing couchdb1               ... done
Removing orderer.example.com    ... done
Removing ca_org2                ... done
Removing ca_orderer             ... done
Removing ca_org1                ... done
Removing network net_test
Removing volume net_orderer.example.com
Removing volume net_peer0.org1.example.com
Removing volume net_peer0.org2.example.com
Removing network net_test
WARNING: Network net_test not found.
Removing volume net_peer0.org3.example.com
WARNING: Volume net_peer0.org3.example.com not found.
---- No containers available for deletion ----
Untagged: dev-peer0.org2.example.com-uavcontract_1-4052ecc3d0172dcbd17e6137ab32bc772e245ae661a1cd090aa00d5401a8d7
Deleted: sha256:2e0dd5cf7d6771e17087c319566cded7c3e7341bddd2df3ea569824da9e690db
Deleted: sha256:5ecfffb741704f337a89de55cd29f4f5a53d97a890a4927edb6185f72a5a1803
Deleted: sha256:040630ed74ceb6e116d554112acac244fd551b66afe44f18e4f4bd5b3576e41c
Deleted: sha256:8ffda67579adf6c317af87429e0f90c742fc22f1c3e23f58c1275736aa33211c
Untagged: dev-peer0.org1.example.com-uavcontract_1-4052ecc3d0172dcbd17e6137ab32bc772e245ae661a1cd090aa00d5401a8d7
Deleted: sha256:1ca5e2ddd34a01523de58aaca17a8740dd478d16a9f0d6d9399f65f9ab00adb8
Deleted: sha256:6c04ceb0607289374080e1ec9f3e80c960aaa06037b15860563e8084b4a82d56
Deleted: sha256:5849ed745c0f005ab45f067a63880581e9e7533ef563fd42532c680c45bab872
Deleted: sha256:8ac6b553f3da21c1a54069291b656dab0281af42613c976ab4961206705c22c8

# remove stopped containers
docker rm $(docker ps -aq)
docker ps -aq
```

Figure 16. Successful Shutdown of Blockchain Network (N)

4. Basic Network Timing Data

The table below annotates various local network timing data points using a 2015 Apple® MacBook Pro® laptop with a 3.1 GHz Dual-Core Intel i7 and 16 GB of RAM running the latest macOS® operating system.

Network Action	Time Units
Blockchain Network Startup/Standup	2 min 19 sec
Client Application Authentication	19.4 sec
O-level DLR Issue Transaction	18.3 sec
D-level Acceptance/Receipt ACK of DLR	17.8 sec
D-level Reissue of RFI DLR	18 sec

Table 2. Network Timing of Various DLR Transactions on Network (N)

The codebase for this presentation is located on the on the Naval Postgraduate School's GitLab server (Norbraten, 2020).

LIST OF REFERENCES

Banerjee, Arnab. (2018). Blockchain Technology: Supply Chain Insights from ERP. 10.1016/bs.adcom.2018.03.007.

Bonanni, Leonardo. (2011). Sourcemap: eco-design, sustainable supply chains, and radical transparency. ACM Crossroads. 17. 22-26. 10.1145/1961678.1961681.

Ethereum. (2020). Ethereum is a global, open-source platform for decentralized applications. https://ethereum.org/en/

Everledger. (2020). Meet the Everledger Platform. https://www.everledger.io

Enterprise resource planning. (2020, December 22). In *Wikipedia*. https://en.wikipedia.org/wiki/Enterprise_resource_planning

Fosso Wamba, Samuel & Queiroz, Maciel. (2020). Blockchain in the operations and supply chain management: Benefits, challenges and future research opportunities. International Journal of Information Management. 52. 102064. 10.1016/j.ijinfomgt.2019.102064

GitHub, Inc. (2020). Repository for the open-sourced Hyperledger Fabric Framework source code. https://github.com/hyperledger/fabric#releases

Hyperledger™. (2020). Open, Proven, Enterprise-grade DLT. https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

Kahn, F. (2019, February 14). *What are the different types of DLTs & how they work?* Data Driven Investor. https://www.datadriveninvestor.com/2019/02/14/what-are-the-different-types-of-dlts-how-they-work

Korpela, Kari & Hallikas, Jukka & Dahlberg, Tomi. (2017). Digital Supply Chain Transformation toward Blockchain Integration. 10.24251/HICSS.2017.506.

Kshetri, Nir. (2018). 1 Blockchain's roles in meeting key supply chain management objectives. International Journal of Information Management. 39. 80-89. 10.1016/j.ijinfomgt.2017.12.005.

Littoral combat ship. (2020). In *Wikipedia*. Retrieved December 3, 2020, from https://en.wikipedia.org/wiki/Littoral_combat_ship

Nakamoto, Satoshi. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

Natarajan, Harish & Krause, Solvej & Gradstein, Helen. (2017). Distributed Ledger Technology and Blockchain. 10.1596/29053.

Network Optional Warfare. (2020). In *NPS Wiki Spaces*. Retrieved January 13, 2020 from https://wiki.nps.edu/display/NOW/Network+Optional+Warfare

Norbraten, T. (2020). Developer code hosted on the Naval Postgraduate School's GitLab repository for this NRP project. [Online]. Available: https://gitlab.nps.edu/tdnorbra/blockchain-mergence

Queiroz, Maciel & Fosso Wamba, Samuel. (2018). International Journal of Information Management Blockchain adoption challenges in supply chain: An empirical investigation of the main drivers in India and the USA. International Journal of Information Management. 46. 70-82. 10.1016/j.ijinfomgt.2018.11.021.

Shahaab, Ali & Lidgey, B. & Hewage, Chaminda & Khan, Imtiaz. (2019). Applicability and Appropriateness of Distributed Ledgers Consensus Protocols in Public and Private Sectors: A Systematic Review. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2904181.

Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, 2017, pp. 557-564, doi: 10.1109/BigDataCongress.2017.85.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943

4. Naval Research Program (NRP) Management Office
   Naval Postgraduate School
   Monterey, CA 93943

5. CAPT Eric Morgan, USN
   DCNO (Fleet Readiness & Logistics) (N4)
   Washington, DC